EXPERT SYSTEMS AS JOB TRAINING AIDS

Marie A. Bienkowski

Information Sciences and Technology Center
SRI International, Menlo Park, California 94025

Abstract

Expert systems are increasingly being used as expert job aids. This paper presents a proposal for development of a general training aid that can be used with expert job aids to enhance on-the-job training. The training aid is based on a goal-directed architecture and can interact in several ways with a trainee, either passively, actively, or responsively. Our focus for an initial test of the training aid will be procedure training based on an expert procedural reasoning system.

1. Introduction

Acquisition of knowledge of procedures is part of many types of instruction, but is perhaps most prevalent in maintenance training. Students learning maintenance must learn procedures for operating, diagnosing, and repairing devices or systems. Most work in maintenance training has focused on device or system simulation ([5]), with good reason: having students work on real devices or systems can be expensive and possibly hazardous. Simulations allow students to explore a complex system's behavior in detail because the simulation's designers can depict relevant inner functioning of the system (e.g., [9]). Furthermore, coupling the simulation with an intelligent tutoring system can provide training in troubleshooting; the tutoring system can introduce faults into the simulation and then monitor a student's attempts to fix them. The tutoring system can detect a student's mistakes and possibly her misconceptions by using expert knowledge of how it would deduce and repair the fault itself.

Training from simulations emphasizes giving the student an understanding of a target system and is primarily designed for classroom use. However, creating simulations requires an extensive investment of time and effort, given that their use is so constrained. As an alternative to this type of training, we propose to investigate ways of moving instruction out of the classroom and onto the job, relying on knowledge engineering already performed for expert reasoning systems.

Expert reasoning systems are increasingly being adopted for use as expert job aids ([13],[18]). In this capacity they may offer problem-solving suggestions rather than ultimate solutions, and, more importantly, they may offer justifications for their suggestions. Recent work in expert systems has emphasized the development of explanation capabilities, some in the context of reassuring users ([15],[3]) and others for tutoring purposes ([4],[14]). Our interest is in coupling articulate expert systems like these with training aids to produce on-the-job training aids - expert job aids that teach as they assist.

Experience suggests that novices can learn procedures for some jobs by relying on manuals and mentors for assistance, applying procedures from manuals when they can and asking a mentor for help when no procedures apply. Over time, the novice who is able to become an expert (and perhaps either has knowledge of a related system or has had training in understanding the target system, for example, with a simulation) builds an internal conceptual model of a system. This model is based on repeated encounters with and numerous observations of the system's behavior under a variety of faults. Common procedures will become rote; more unusual ones may lead to knowledge of previously unexplored aspects of the system or serve to point out misconceptions the novice might have in understanding.

The observation that on-the-job learning occurs naturally as described above suggests that expert job aids could prove useful for on-the-job training. However, simply adding an explanation component onto an expert system to allow it to articulate its knowledge and reasoning does not automatically produce a job training aid. Instead, strategies for job training must be devised, implemented, and empirically tested. The focus of these strategies must be to help a novice perform a job while simultaneously facilitating learning. As it turns out, the same components found in intelligent tutoring systems can be used in on-the-job training aids: a model of expertise, a student model,
training strategies, and a user interface.

In the remainder of this paper, we present a proposal for creating a training aid that can be used with an expert job aid. In the following section, we discuss the types of interaction that could occur between the training aid and a user. In Section 3, we describe a general architecture for the training aid. Our main concern in designing this architecture was to provide flexibility in the way that the training aid interacts with a novice to enable us to test different strategies for on-the-job training. In the fourth section, we describe some strategies for procedure training. In Section 5, we briefly describe a procedural reasoning system developed at SRI [7], called PRS (Procedural Reasoning System), which will be used in the initial test of the training aid. PRS is an existing expert reasoning system that can represent and manipulate structured, procedural knowledge. Before concluding, we discuss some specifics of our approach to procedure training using the training aid with PRS.

2. An Evolutionary Approach to Building a Training Aid

Our approach to training using expert systems has been influenced by examination of two types of expert systems: diagnosis/repair systems and routine design systems. These are of interest for several reasons. One is that they can be applied to diverse domains, thereby extending the usefulness of our training aid. Each has been investigated sufficiently so that systems for solving real problems in these areas have been developed. The systems developed attempt to mimic problem-solving principles that are used by human experts, making them highly suitable for training purposes. Furthermore, each type relies on very different kinds of knowledge, which provides a good test for explanation capabilities.

A simplistic view of a training aid is as a question-answering (Q/A) component attached to an expert system. For example, a novice may be presented on the job with a problem that needs to be diagnosed. She then runs a diagnosis expert system and feeds in data as the program requests it. Whenever the expert system asks a question, she can respond with a question herself (assuming that time is not a critical factor). Typical questions from a novice would pertain to why the current question was asked by the expert system (i.e., what is the state of reasoning (what hypotheses does it have, and which have been ruled out?), what other kinds of diagnoses are possible, and how might the hypotheses change if different data were used (i.e., "What would happen if..." questions). In this case, the novice is taking the initiative to respond to the system's questions and investigate the knowledge that led to those questions.

We call the style of training described above "passive." In passive training, the novice's questions initiate goals which the training aid satisfies by producing explanations. This simplifies management of the training discourse because the discourse initiative (roughly, the power to bring up a new topic) remains with the novice, leaving the training aid passive except when it is asking questions to gather data. One potential problem with passive training is that the question-asking behavior of the reasoning system may not be sufficient to elucidate the model underlying that behavior. If a novice can ask questions only when the system asks a question, unreported processing may not be noticed by unmotivated novices. A more serious problem is motivating a novice to ask questions in the first place.

The problem of unreported processing is exacerbated in expert systems that have a "design style" of interaction. For the most part, all the relevant parameters and constraints for a design are known ahead of time. These are presented to the design expert system, which then operates silently until the result is produced. The only questions that might be asked during problem solving would deal with constraint satisfaction (e.g., determining which constraints to relax when an impasse is reached). Apart from these points, observable behavior arises only when the design is complete, and intermediate results must be accessed through a backtrace of the problem solving. For such laconic problem-solving systems, the training aid should take the initiative to present information before it is specifically requested (partly based on information found in a student model). We call this an "active" training method. An active training aid for design could report high-level design goals as they are pursued. For diagnosis, it would present intermediate hypotheses and results in between the times that the expert system is requesting values for problem parameters from the novice.

The architecture we describe in the next section can be used to implement either active or passive training. It can also be used for combining the two into a training method we call "responsive." Passive training requires a question-answering capability that responds to the novice's stated goals. Active training requires managing goals that direct the training aid to provide relevant information during problem solving. Their combination, responsive training, requires a method of switching between passive (to keep out of the way of the knowledgeable, motivated novice, or for use in time-critical situations) and active (to report important events or help along less successful novices). Our basic architecture allows for an evolution from passive to active to responsive, such that each stage builds on the technology developed for the preceding ones. Evolutionary implementation is an important part of our approach to building complex training aids.
3. A Generic Training Aid Architecture

It is widely recognized that for an intelligent program to be responsive, its behavior must be directed by the achievement of explicitly represented goals. The goal-directed architecture selected as the basis of our generic training aid is an extension of a task management system designed to control an explainer for problem-solving systems [1]. Its precursor is the NASL system [11], from which it adopts terminology and ideas about control, as well as its view that actions are a series of interleaved planning and execution steps. The architecture is being extended to incorporate some ideas about reactivity from work on procedural reasoning [7]. Mapping goals to plans, instantiating plans to create tasks, then achieving goals by executing tasks will provide the basic method for controlling training in a variety of situations. Plans will be followed to control the training discourse. They represent ways of achieving goals like “explain procedure” or “outline steps.” (Plans like these are called “conversational move classes” in [16].)

The task execution manager, called TaskMan, provides a general and flexible control mechanism for pursuing hierarchical plans by managing tasks. TaskMan has several useful features. First, it allows that there may be several plans for achieving one goal, and plans can be invoked by name or by a pattern representing the desired goal. Second, the plans it uses are representationally adequate for many purposes. They have been used for describing the actions of an explanation program, a route planner, and an academic counselor [2]. Third, tasks, which represent the active parts of plans, are preemptable and can be suspended pending completion of some other task. For example, a task for explaining can be suspended to service one for answering a question. It is also possible to create both a central set of low-level plans that are independent of the particular type of training (e.g., plans for answering “why” type questions), and higher-level plans for controlling specific types of training.

The components making up a training aid are shown in Figure 1. Input to the training aid will be handled by a general input mechanism. To simplify requirements on this mechanism for our proposed training aid work, we require that the novice's goals be explicitly stated in, or directly inferable from, questions. To enforce this, we envision the input mechanism as a set of menus from which the novice can construct questions. Naturally, the menus will present options that reflect the current situation. ¹

Goal achievement begins with the goal mapper part of TaskMan which maps goals onto plans (these may also be called schemas or procedures) that will achieve them. Conflicts in mapping (i.e., one goal being satisfiable in more than one way) are handled by reasoning about the choice among several goals (currently, a simple back-chaining reasoner is used). If a plan cannot be found to satisfy a goal, the goal mapper uses the reasoner to produce a restatement of the goal into subgoals such that each subgoal can be achieved.

Plans pursued for training are grounded in actions that deal with presenting information to the novice. These actions will access expert knowledge bases to obtain the content to be presented. We classify the types of knowledge needed for training as organizing (how the knowledge is structured) and supporting (what the rationale is for the knowledge). The organizing and supporting knowledge bases will be accessed by the message generator which matches a symbolic description of the information desired onto actual data from the expert system. Once a message is constructed, a language generation component will express it. The message generator, then, accesses the knowledge bases to determine the “what to say” part of a message, leaving the “how to say it” to the language generation component.

This basic system is sufficient for passive training. In passive training, we assume that the expert system runs in its normal mode of operation and asks questions of the novice at certain points, using whatever output mechanism it normally does. Instead of answering the question, however, the novice can indicate that she wants to ask a

¹We adopt this minimal design because we do not wish to handle the problem of inferring goals from natural language input, and we want to constrain the goals that the training aid is requested to achieve to avoid problems associated with planning actions to achieve novel goals. However, our design does not preclude use of a better language-input mechanism.
question. At this point, the menu-based input mechanism we described would be invoked, and the training aid would pursue tasks to answer the question. When it was done, control would be returned to the expert system. While the dialogue transitions would not be smooth, this represents a reasonable first step in developing a responsive training aid and also produces a usable question-answering system that can be tested with real users.

To explicate fully the model that the expert system embodies, the training aid must be capable of running in a more verbose, active mode. In this mode, the training aid will be monitoring the actions of the expert system and explaining important actions as they are performed. This requires that the training aid have control of the dialogue (instead of the expert system) and that the training aid generate its own goals (instead of the novice initiating them) based on the actions of the expert system. The move from passive to active, then, requires a new set of goals (and plan to achieve them) and a shift in initiative from the user to the training aid.

The training aid must watch for events that look interesting enough to be reported to the user, basing its decisions on a model of the user. The requirement, plus the need for a supporting database, will constrain the type of expert system that is suitable for use with this training aid. In particular, it requires that the expert system have a global working memory, so the training aid can monitor its behavior, and a means for associating justifications with rules, if justifications are not provided.

The final step in the evolution of the training aid involves adding a metalevel process that can control the focus of attention among the expert system, the active explainer system, and the passive Q/A system. This metalevel process, which is a discourse manager, must evaluate goals from four sources: (1) the user, (2) the expert system, (3) the question-answering system, and (4) the explainer. Design of this process will await the development of the passive and active modes of operation.

4. Strategies for Explaining Procedures

In any mode of training operation, a primary training strategy is tailoring explanations to help a novice understand how both specific knowledge of the target system and general principles can be applied to problem solving. Thus, this is our initial focus for procedure training. Such tailoring requires that we identify the important features of procedures that we want a novice to learn and see that these are emphasized by the explanations given. The second step (not described in this paper) will be to integrate this with a model of the novice's knowledge.

One crucial feature of a procedure is what task it can be used for, or what goal is achieved by its application. It is important to focus on this goal while the procedure is being used to perform a task or is being explained. Focusing on a specific goal can be important in helping a novice organize and retain material.

Another feature is the specific conditions under which the procedure can be used. Goal achievement may not be sufficient to determine a procedure's applicability; other conditions may also need to be satisfied. Some of these conditions may be a consequence of the way the procedure operates, others may reflect strategic metalevel knowledge about procedure application (such as doing simpler tests first). These conditions must be explained after the goal of the procedure is understood.

Procedures are composed of organized steps that combine to achieve a goal. In explaining the internal structure of a procedure, it is important to emphasize what each step contributes to the overall plan, i.e., what its purpose is. The part of the plan being explained might be a direct step toward achieving the goal; it might be a preparatory condition; or it might be an information-gathering step. The purpose of each step should be related, where appropriate, to the internal conceptual model of the device the student is assumed to be creating ([10]). This could be done by referring to a functional or structural model, and ensures that the level of justification of the step corresponds to the most understandable level of explanation. Finally, it is important for the novice to learn how the steps combine (causally, temporally, etc.) to achieve the goal.

Explaining procedures requires that the expert reasoning system represent its procedural knowledge in an explicit fashion. In the next section, we describe in more detail the PRS system, which provides a rich declarative representation of procedural knowledge.

5. Procedural Reasoning System

PRS is a tool for interpreting and reasoning about procedural knowledge that is represented in declarative form. It has the interesting property of being both goal-directed and reactive; that is, it can follow procedures to achieve goals, but when new information is discovered, it can preempt its current operation to follow procedures relevant to the new information. PRS interprets operators that can be used to represent achievement of arbitrary behaviors, state maintenance, and inference of new facts. Factual information (maintained in a database), goals, and procedure steps are represented using these operators and predicate calculus formulas. Procedures, called knowledge areas or KAs, are composed of an invocation part, consisting of the goal achieved by this procedure and applicability conditions, an effects property, which lists states that are true if the pro-
procedure succeeds, and a body, containing the steps of the procedure. Metalevel procedures are used to determine which procedures are best to pursue when more than one applies. PRS also has a facility for managing individual agents, each containing its own procedural knowledge and factual database, that can cooperate to achieve goals.

PRS relies on a graphical manipulation system called GRASPER II [12] for creating and displaying its procedural knowledge. GRASPER II is a general graphical display tool which can also be used for information presentation for the training aid. PRS has been applied to control of the space shuttle's reaction control system (RCS) [6]. This application includes a simulation of the behavior of the reaction control system, a database containing a structural and functional description of the RCS, an interface to the simulator, and a set of KAs corresponding to the maintenance methods for the RCS and its subsystems. PRS was developed for reasoning about and executing procedures, so it has explicit representations of procedural knowledge. It also has a means for associating justifications with each KA, providing the necessary input for the training aid.

In the previous section, we listed four areas that were important to emphasize for procedure understanding: focusing on the goal, explanation of the applicability conditions, justification of each procedure step, and making clear the combination of the steps. We discuss strategies that will be used by the training aid to teach each of these in turn.

PRS displays the goal it is currently working on in predicate calculus form when it is running in a debugging mode. For training, an English form is obviously more suitable. The training aid, when it chooses to focus on a procedure for explanation, will cause a translation to be done by the language generator. Both the goal and the applicability conditions for relevant procedures will be displayed in English. However, the training aid must determine the relevance of parts of the applicability condition before generation. For example, some parts of the condition simply bind variables for the rest of the condition or body of the procedure. In explanations, these parts turn into complex noun phrases that are generated wherever the variable is used. Such assertions should not be mentioned as part of the applicability condition. Hence, strategies for determining the relevance of parts of a procedure will be utilized in the plans that translate the applicability conditions into English.

An important part of expert knowledge is the set of strategies used to select a problem-solving approach. Such strategies appear in PRS as constraints on the invocation of procedures as implemented by metalevel KAs. When a metalevel KA is used to select between two slightly different procedures, the reason for the choice of one over the other must be articulated. For example, one metalevel KA is used to ensure that safety procedures are followed before any others. The concept of ensuring safety, then, will be explained when first encountered.

PRS, using GRASPER II, can display KAs in graphical form (see Figure 2) as the steps in them are followed. Such a display seems useful for making the connection between steps explicit and can be exploited by the training aid. As in a transition diagram, steps are labeled with behaviors whose successful execution causes transition to the next state. As before, the predicate calculus forms labeling the edges need to be translated into English, after filling in the values of variables (indicated by "$" in Figure 2).

While the ordering of steps can be represented explicitly in graphical form, it does not make explicit why a given ordering was selected by the procedure's designer. In cases where there is a simple causal connection between two steps (e.g., after a decision node), presenting an edge labeled with a sentence will be sufficient. In other cases, ordering may reflect safety conditions or ease of maintenance, so the training aid will point out the justification for the ordering of steps. This information will be added as part of the supporting knowledge base for the RCS application.

The final important part of procedure training is defining the purpose of each step. This enables the user to
see why a KA was constructed as it was, leading to better understanding and retention. In PRS, the purpose of each step is not explicitly part of the representation of a procedure in a KA and may not be immediately obvious. For example, sometimes the test operator (?) is used to obtain a value for a variable, but other times it is used to implement a test and branch function. For this particular example, a rule can be written for determining the purpose of the step; in other cases, the purpose will have to be represented explicitly. Presenting this information without becoming overly verbose will be done by reliance on the user model.

In Section 2, we described three modes of training aid operation: passive, active, and responsive. For a passive training mode, we need only integrate a question answering interface with PRS. When PRS asks a question, the user can ask "why." Answers to "why" questions can be given as a graphical presentation of the relevant procedures, as described above. Questioning can continue with the user asking to see justifications for individual steps or their ordering by clicking on the questioned part of the procedure.

Active and responsive training require monitoring the actions of PRS coupled with tests of relevance of information. Monitoring will be done by examining PRS's state after each reasoning step. Processing will be interrupted for presentation of relevant information before or after the fact. For example, by monitoring PRS, the training aid could interrupt whenever an action is performed. Then, the action taken, if relevant, can be justified. This is, in effect, "looking back" and tells what PRS did to motivate the action. Alternatively, the training aid could "look ahead" to explain anticipated actions. This would result in a description of paths that might not be followed. Both of these strategies will be provided to allow their relative usefulness to be tested. This demonstrates how the training aid presents opportunities for testing different training strategies.

6. Summary

In this paper, we have presented a proposal for developing an expert job training aid that employs strategies for training while aiding a novice with a task. We have also presented a specific application of this training aid for procedure training. We have described a goal-directed architecture for a training aid that will enable an evolutionary development of capabilities. Building on the simplest capability, passive training, a fully responsive training aid can be realized. Furthermore, the goal-directed design will allow us to explore various ways of determining relevance and presenting information. For example, there is currently no connection between the procedural knowledge and a schematic representation of the system. Plans that achieve the goal of justifying a step in a procedure could be modified to utilize a schematic for explanation when one is available. Overall, we expect that our approach will encourage the use and further development of expert job aids for training.

7. References


Artificial Intelligence Center, SRI International, Menlo 
Park, California. January.

Technology for the Maintainer's Associate.” AFHRL- 
TR-86-31. Human Resources Laboratory, Brooks Air 
Force Base, Texas. December.

“Intelligent Tutoring Using the Integrated Diagnostic 
Model: An Expert System for Diagnosis and Repair.” 
Expert Systems in Government Symposium. McLean, 
Virginia. October. pps. 128-135.

ing Expert Systems More Like Experts.” Expert Sys- 
tems, Vol.4, No. 3, August.

Discourse Conventions in Tutoring.” Expert Systems 
145-156. October.